**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

# Дипломна робота

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»**
**спеціальності 122 «Комп'ютерні науки та інформаційні технології»**

**на тему: «Підсистема оплати послуг»**

Виконав (-ла):
студент (-ка) IV курсу, групи ТМ-61
Клименко Дмитро Олексійович _____

Керівник:
Професор кафедри АПЕПС, доктор економічних наук
Сігайов Андрій Олександрович _____

Рецензент:
Ст. викладач кафедри АЕС і ІТФ, кандидат технічних наук
Воробйов Микита Валерійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

**Національний технічний університет України**
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки 122 Комп’ютерні науки та інформаційні технології
Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

# ЗАВДАННЯ

**на дипломну роботу студенту**

_____Клименко Дмитро Олексійович_____
(прізвище, ім’я, по батькові)

1. Тема роботи _Підсистема оплати послуг_____
_____
керівник роботи Сігайов Андрій Олександрович, професор, д.е.н_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _19.06.2020 року_____
3. Вихідні дані до роботи _____
_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _створити підсистему оплати послуг; підключети до системи оплати послуг; створити адмін-панель керування базою даних в підсистемі;_____
_____
_____

5. Перелік ілюстративного матеріалу
_Фінальний вигляд продукту; принцип роботи підсистеми; принцип роботи бази даних; середовище написання;_____
_____
_____
_____

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |

7. Дата видачі завдання "_8_" __січня_____2020_ р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітки |
|---|---|---|---|
| 1. | Затвердження теми роботи | 08.01.2020 | |
| 2. | Вивчення та аналіз задачі | 03.03.2020 | |
| 3. | Розробка архітектури та загальної структури системи | 02.04.2020 | |
| 4. | Розробка структур окремих підсистем | 12-30.04.2020 | |
| 5. | Програмна реалізація системи | 07.05.2020 | |
| 6. | Оформлення пояснювальної записки | 15-25.05.2020 | |
| 7. | Захист програмного продукту | 07.06.2020 | |
| 8. | Передзахист | 08.06.2020 | |
| 9. | Захист | 19.06.2020 | |

Студент _____ ____Клименко Дмитро Олексійович__
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _ __Сігайов Андрій Олександрович___
(підпис) (прізвище та ініціали,)

# АНОТАЦІЯ

Головною метою роботи є створення безпечної та зручної у повсякденному використані платіжної підсистеми послуг. Яка надає змогу легкого переходу середньому та малому бізнесу для роботи в онлайн, зі зручним інтерфейсом керування та роботи з базами даних.

Підсистема оплати послуг підключена до однієї з найбезпечніших та популярніших систем, а саме LiQPay від ПриватБанка, що дозволяє підсистемі проводити транзакції без переходів.

Даний вид оплати дозволить компаніям перейти в онлайн режим, продавати свої послуги чи товар, автоматизувати процес замовлення та оплати в месенджері, Даний Telegram-bot вирішить проблему компаній малого та середнього бізнесу, так як не потрібно буде створювати сайт для переходу в online, консультувати людей в режимі дзвінка.

Загальний обсяг роботи: 48 сторінок, 26 ілюстрацій та 4 бібліографічних найменувань.

Ключові слова: підсистема оплати послуг, онлайн платежі, транзакція, Telegram-bot, платіжна система.

# ABSTRACT

The main purpose of the work is to create a secure and convenient in everyday use payment subsystem services. Which allows easy transition of medium and small businesses to work online, with a user-friendly management interface and work with databases.

The service payment subsystem is connected to one of the most secure and popular systems, namely LiQPay from PrivatBank, which allows the subsystem to conduct transactions without conversions.

This type of payment will allow companies to go online, sell their services or goods, automate the ordering and payment process in the messenger. This Telegram-bot will solve the problem of small and medium-sized businesses, as it will not be necessary to create a site to go online in call mode.

Total volume of work: 48 pages, 26 illustrations and 4 bibliographic titles.

Keywords: service payment subsystem, online payments, transaction, Telegram-bot, payment system.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ПК — персональний комп’ютер.

СКБД — система керування базами даних.

БД — база даних.

СМБ — середній та малий бізнес.

# ВСТУП

В 2020 році існує багато шляхів оплати послуг, сучасні компанії повинні підтримувати максимальну кількість способів, як в offline, так і в online режимах. На даний момент, є багато способів оплати в online, і не існує єдиного ідеального, чи більш популярного способу. Хтось проводить платіж за послуги через мобільний додаток банку, інший через сайт компаній за допомогою платіжних систем, та моя мета розробити Telegram-bot за допомогою якого, компанія може виставити рахунок в месенджері Telegram, а користувач матиме змогу оплатити рахунок в один клік.

Даний вид оплати дозволить компаніям перейти в онлайн режим, продавати свої послуги чи товар, автоматизувати процес замовлення та оплати в месенджері, а месенджери є найбільш зручним способом спілкування, тому їх використовують дуже часто, по декілька раз на день. Даний Telegram-bot вирішить проблему компаній малого та середнього бізнесу, так як не потрібно буде створювати сайт для переходу в online, консультувати людей в режимі дзвінка.

Для побудови Telegram-bot було використано  Telegram-APIs, у якості системи контролю бази даних було вирішено використовувати SQLite. Основним модулем для розробки Telegram-bot буде використовуватися бібліотека pyTelegramBotAPI реалізована на мову програмування Python, а для зручної роботи з базою даних SQLite – модуль Flask-SQLAlchemy.

Взаємодія з розробленою системою проходить у режимі реального часу, будь які зміни, оповіщення тощо власник буде отримувати миттєво.

# 1 ЗАДАЧА СТВОРЕННЯ ПІДСИСТИМИ ОПЛАТИ ПОСЛУГ В МЕСЕНДЖЕРІ TELEGRAM

В даний момент часу, майже всі компанії працюють в режимі онлайн, кожна з них бажає отримати максимальну кількість способів та можливостей контакту з потенційними клієнтами. Також, з кожним роком мобільні телефони охоплюють більше користувачів чим персональні комп'ютери, а найпопулярнішими мобільними додатками стають соціальні мережі та месенджери, саме тому гостро постає питання просування своїх товарів чи послуг за допомогою месенджерів.

В першому розділі розкривається суть дипломної роботи та поставленої задачі, завдання яке повинне бути виконане. Описані вимоги, актуальність, проблеми та мета самої підсистеми.

## 1.1 Суть створення підсистеми оплати в месенджері

Метою цієї дипломної роботи є створення бота, за допомогою якого компанії зможуть з легкістю контактувати з клієнтами в Телеграмі, демонструвати свої товари, та проводити транзакції.

Дана задача обумовлює такі етапи роботи:

— створення бази даних для продукції чи послуг;

— наповнення розробленої БД інформацією про товар/послугу, інформацію про них (фото, опис, ціна та ін.);

— створення підсистеми для проведення онлайн-транзакцій;

— інтеграція з платіжною системою;

— розробка програмного продукту з зрозумілим інтерфейсом користувача.

Програмний продукт передбачає фунціонал для демонстрації окремих категорій товарів чи послуг, детальної демонстрації кожного з них та проведення онлайн оплати. Завдяки йому користувач зможе ознайомитись, обрати та придбати один з представлених товарів чи послуг. Програмний продукт повинен:

— ідентифікувати категорію, товар, кількість, що вибрав користувач;

— вивести інформацію щодо фінальної вартості замовлення;

— надати можливість здійснити оплату за замовлення;

— дозволити власникам зручно керувати БД та відслідковувати всі замовлення.

## 1.2 Вхідні дані

Розроблена база даних зі зручним інтерфейсом її заповнення та ведення. Відповідно, вхідними даними для цієї бази даних є реальні товари чи послуги, які заповнюють власники.

Від користувача, на початку, підсистема не потребує додаткових даних. Для роботи потрібно відкрити створеного телеграм-бота, обрати потрібні йому послуги чи товар, їх кількість, та тільки потім ввести дані платіжної карти для проведення транзакції. Тому робота з підсистемою дуже проста і не потребує ніяких спеціальних навичок.

Вхідними даними є інформація про всі товари, а саме: фото, детальний опис, ціна за вагу чи штуку, наявність та варіаційний ряд складу.

Завдяки невеликій кількості вхідної інформації розроблюваний телеграм бот зовсім не викликає проблем у користувача під час його використання та створює необхідне зручне середовище для власників для демонстрації та продажу свого товару.

## 1.3 Компоненти підсистеми

Головними частинами, що забезпечують працездатність підсистеми з завдання дипломної роботи є:

— база даних;

— інтегрована підсистема оплати.

Розроблена база даних надає користувачеві інформацію про наявний товар компаній, у той час як підсистема допомагає проводити транзакції не покидаючи мобільного додатку.

Використання створеної БД забезпечується сімейством програм SQLAlchemy.

## 1.4 Актуальність підсистеми

Підсистема, яка розроблена в процесі виконання бакалаврської дипломної роботи є корисною для компаній, яким потрібний спосіб просування та продажу товарів через месенджер. Підсистема не потребує у користувача професійних навичок володіння ПК чи мобільним пристроєм, тому може бути легко використана. Необхідна для багатьох середніх та малих бізнесів в Україні.

# 2 АНАЛІЗ ПРОБЛЕМИ СТВОРЕННЯ ПІДСИСТЕМИ ОПЛАТИ ПОСЛУГ В ТЕЛЕГРАМ

Першочерговою задачею є підключення до системи оплати, структуризація даних та створення зручного відображення товару з бази даних. Отримані дані з бази даних повинні чітко відображатись та бути актуальними, платіжні дані клієнтів не фіксуються та не зберігаються підсистемою, а мають переходити до платіжної системи.

## 2.1 Платіжна підсистема

Онлайн оплати одне з найскладніших питань сьогодення, існує безліч способів її проведення, безліч правил та установ щодо систем та підсистем оплати. Саме тому, найважливішим питанням постає платіжна система до якої буде підключено підсистему, та через яку будуть проходити транзакції. За весь період свого існування ПриватБанк здобув довіру та досягнув найбільшого успіху в сфері онлайн оплат та збереження конфіденційних даних, саме тому було обрано LiqPay, платіжну систему ПриватБанка, як систему до якої буде підключений бот чи підсистема оплати.

Платіжна система — платіжна організація, члени платіжної системи та сукупність відносин, що виникають між ними при проведенні переказу коштів. Проведення переказу коштів є обов'язковою функцією, що має виконувати платіжна система.

Протягом майже всього існування людства відбувається процес торгівлі. Існуючі на сьогоднішній день способи не можливо порівняти з тими, що були раніше. Перша платіжна система в Україні була створена в 2001 році, але онлайн оплати

з’явились не так давно, вони охоплюють все більше і більше транзакцій, та особливо актуальні в умовах карантину. Готівка, як і банківські картки починають поступатися цьому способу оплати. Розвиток платіжних систем дав можливість створення підсистем, що дозволяють збільшувати кількість каналів продажу та створювати оплати без переходів на інший сайт чи систему.

### 2.1.1 Принципи та функції підсистеми

Принципами платіжної підсистеми є:

— безпечність;

— доступність;

— комфортність.

Під безпечністю мається на увазі те, що інформація, яка передається до платіжної системи не зберігається, використовується чи передається третім особам. Підсистема представляє функціонал, що забезпечує онлайн передачу платіжних даних без втрат та розголошень.

Доступність обумовлює наявність зручного та безкоштовного доступу. Відсутність реєстрації, особливих умов чи статусу користувача.

Комфортність – це зручний спосіб оплати, без додаткових переходів до системи оплати послуг, реєстрації на сторонніх сервісах, дзвінках чи повідомленнях в банк.

Функціями підсистеми являються:

— демонстрація;

— обробка та аналіз замовлення;

— проведення онлайн сплат.

Однією з найважливіших функцій є демонстрація всього спектру послуг чи товарів, структуризація та керування даними. В даний момент, найбільш важливим для клієнтів є дизайн та інтерфейс.

Телеграм бот повинен не лише вміти гарно відобразити ассортимент, але мати можливості для обробки інформацію про замовлення, категорію, тип, кількість і ціну товару.

В першу підсистема використовується для проведення транзакцій, і вона є дуже зручним інструментом проведення оплати в будь-якій сфері. Підсистема справді можна вважати невід’ємною частиною кожного бізнесу. Справедливо можна сказати, що цей спосіб оплати є одним з найбільш актуальних сьогодні, і зазвичай використовується з легкістю.

### 2.1.2 Структура БД

База даних представляє собою набір даних, які пов’язані між собою спільними ознаками, що створює можливість для обробки автоматичними засобами. Головним в базі даних є її універсальність використання та збереження даних. Тому, жодні зміни будь-яких даних у БД не змушують покращувати програмне забезпечення.

Головні функції Бази даних:

— цілісність даних;

— безперервний доступ до інформації для адміністраторів;

— обмежений доступ для користувачів;

— створення залежності даних;

— можливість редагування БД для адміністраторів;

— швидкість роботи з інформацією та даними.

Для покращення використання та зручної взаємодії з БД використано дві концепції:

— адміністрація бази даних;

— система за допомогою якої відбувається процес керування даними.

Адміністратор має змогу використовувати базу даних не закриваючі підсистему, йому не потрібно переходити в інший додаток, має змогу керувати БД як с ПК, так і з мобільного пристрою.

СКБД являє собою комплекс програм, що надає змогу повного керування над базою даних. Основні функції які можуть виконувати СКБД:

— створення категорій, товарів та замовлень;

— додавати нові записи до таблиці;

— видаляти записи у таблиці;

— повне очищення та створення нової БД;

Система керування базами даних покращує процес роботи з базою даних, надає функціонал доступу та полегшує взаємодію з БД.

Адміністрація бази даних це група чи одна людина, вони слідкують за БД, товарами та замовленнями в ній. Надати права адміністратора може тільки розробник, заповнивши дані імені та номера телефона адміністратора чи групи адміністраторів.

База даних містить в собі всю інформацію про асортимент, замовлення та кількість товару, може бути редагована одним з адміністраторів, повністю видалена та створена знову. Також, є зручний інтерфейс для роботи з замовленнями, а саме адміністратор має змогу переглянути всі замовлення, дані про клієнта, замовлення та сплату, після цього опрацювати та видалити замовлення.

### 2.1.3 Данні в підсистемі

Як було сказано, однією з найважливіших складових підсистем є дані.

Особливість підсистеми полягає в тому, що в одному місці розміщується асортимент товару з можливістю онлайн замовлення та оплати.

Система оплати послуг, на відмінну від підсистеми по функціоналу обмежена тільки проведення транзакцій. А оплати через інші канали неможливі без переходу на сайт платіжної системи.

Дані у підсистемі це реальні товари чи послуг, які користувач може отримати від компанії. Підсистему можна налаштувати на два окремих типу даних:

— товари (матеріальні речі які може отримати користувач);

— послуги (нематеріальні, дії що бажає отримати користувач).

Для відображення кожного з них використовується різна система виводу даних.

## 2.2 Висновки

Підсистема послуг надає можливість демонструвати користувачам товари чи послуги компаній, зручно виводити асортимент по категоріям,  демонструвати товар за допомогою фото, опису.

Важливою функцією підсистеми оплати послуг є процес транзакції, клієнт може використати декілька способів не покидаючи підсистему (додати нову карту або  використати збережену в Apple- чи GooglePay), або перейти в Приват24 для здійснення транзакції.

# 3 ЗАСОБИ РОЗРОБКИ

Для створення підсистеми потрібно організувати базу даних, зробити зручний функціонал та забезпечити швидку роботу. Сам проект написаний на мові програмування Python та використовує бібліотеки Flask, Flask-SQLAlchemy для роботи із базою даних. Фінальний вигляд продукту зображений на рисунку 3.1.

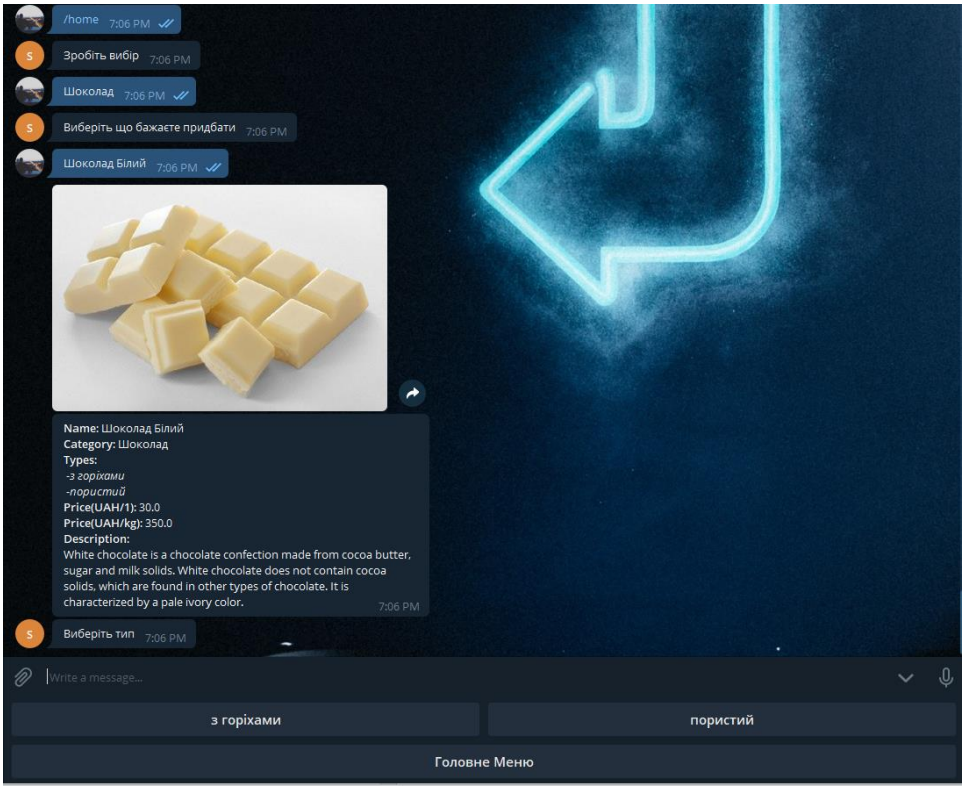



Рисунок 3.1 — Фінальний вигляд продукту

Для зручної та швидкої взаємодії з базою даних була обрана БД MySQL і система керування базою даних SQLite. Роботу з БД не виходячи з підсистеми забезпечує розширення Flask-SQLAlchemy. Саме тому, адміністрація підсистеми, може легко та зручно працювати з інформацією та даними в БД.

## 3.1 Мова програмування Python

Python простий у використанні, повна мова програмування забезпечує набагато більше інструментів для створення та підтримки великих програм, ніж shell. Якщо також порівнювати, це краще для оброблення помилок ніж C і є дуже високорівневою мовою програмування. Також має вбудовані типи даних високого рівня, такі як гнучкі масиви та словники, ефективне застосування яких C вимагає значного часу

З більш загальними типами даних, Python застосовується до більш широкого кола завдань, ніж Awk, і навіть Perl, в той час як багато речей на Python так само просто робляться.

Python дозволяє розділити додатки на модулі, які потім можна використовувати в іншому коді. Python постачається з великою бібліотекою стандартних модулів, які можна використовувати як основу для нових програм або як приклади при вивченні мови. За рахунок них можна відразу з “коробки” працювати із файлами, системними викликами, мережевими підключеннями та навіть інтерфейсами в різних графічних бібліотеках.

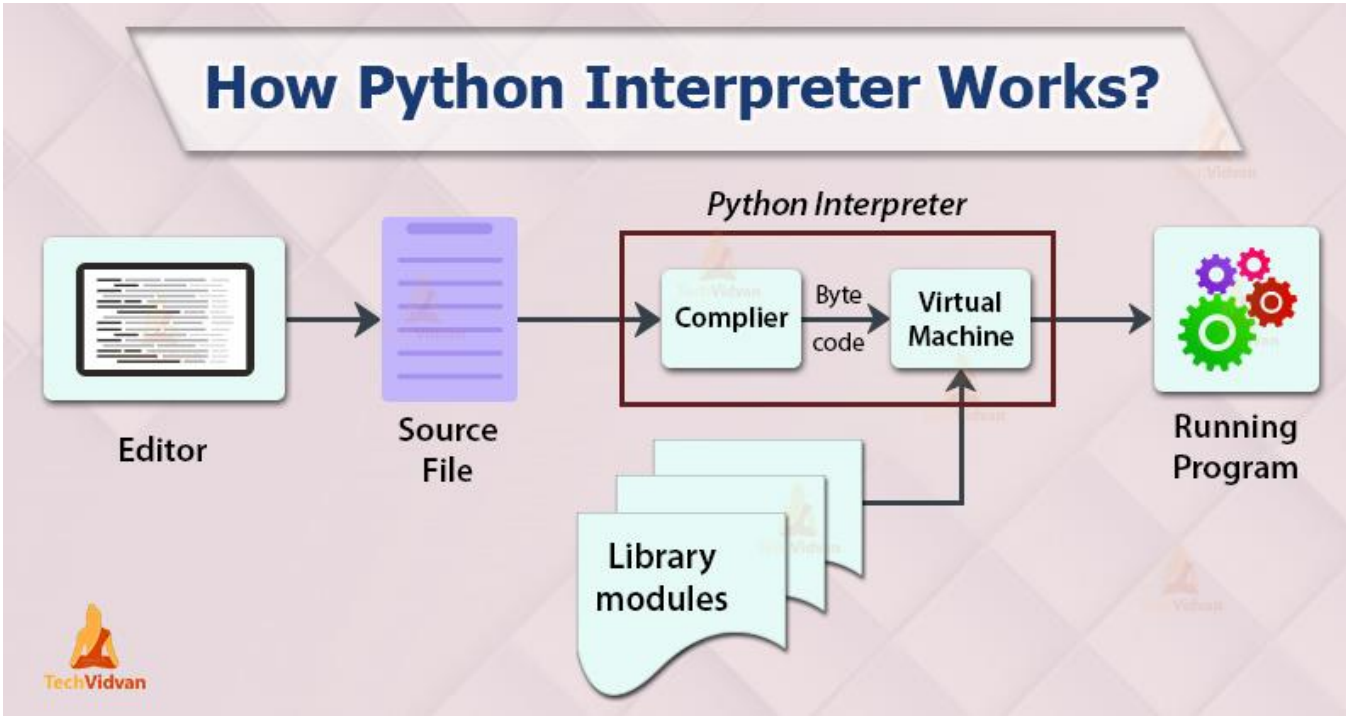



Рисунок 3.2 — Принцип роботи Python

Python є інтерпретованою мовою програмування, яка значно допомагає економити час, який зазвичай витрачається на компіляцію. Також інтерпретатор використовується інтерактивно, що дозволяє експериментувати з мовними можливостями, писати шаблони програм або тестові функції при розробці “знизу-вверх”. Короткий опис показано на рисунку 3.2. Це також зручно як настільний калькулятор. Python дозволяє писати дуже компактною і читабельно будь які програми.

Мова розширення Python: знання C дозволяє додавати нові вбудовані функції, або модулі для виконання критичних операцій з максимальною швидкістю або для інтерфейсу в комерційних бібліотеках, доступних тільки в двійковому форматі. Інтерпретатор Python може бути інтегрований в програму, написану на C. Програми, написані на Python, зазвичай значно менше еквівалентні C або C++ з кількох причин:

- Типи даних високого рівня дозволяють вам виражати складні функції в Директивам.
- Інструкції з групування виконуються за допомогою відступів замість дужок.
- Відсутність необхідності рекламних змінних;

Python в даний час використовується десятками тисяч розробників по всьому світу, і число людей, що використовують його стрімко зростає з геометричною прогресією. Python залучає користувачів з різних причин. Він використовується для розробки програмного забезпечення і дозволяє розвитку набагато швидше, ніж звичайні, традиційні мови Java, C або C++. Ця мова працює так само добре в Windows як і в інших операційних системах як UNIX, Macintosh і OS/2, які може бути використаний для зручного розгортання невеликих програм або скриптів і розробки великих програм. Python може надати доступ до потужного та легкого у використанні 29 набору інструментів GUI. Традиційний тип C і Паскаль машинної мови мають кілька характеристик, наприклад, жорсткі типізація, базові формули, складні (і, звичайно, великі) цикли, і потреба у великих кількостях кодів для виконання відносно

дрібних завдань. Якщо взяти Java зараз є досить новим, але розподіляє більшість функцій, включених до цього списку. Роботу з Python значно полегшує сувора типізація з погодяться всі програмісти знайомі з традиційними мовами.

Якщо порівняти Python до інших мов програмування

Є багато різних відмінностей Python, ми перераховуємо основні:

1. Керування пам'яттю повністю автоматичне — вам не потрібно турбуватися про призначення або відпускання пам'яті. Немає загрози "небезпечного посилання". Єдиною мовою, яка пропонує таку концепцію є Java.
2. Типи, зв'язані з об'єктами, а не змінними, тобто виходить, що значення будь-якого типу можна призначити змінної, і що (наприклад) таблиця може містити різні типи об'єктів. Традиційні мови не надають цю функцію.
3. Виконання операцій на більш абстрактному рівні.


Рисунок 3.3 — Переваги Python

Ось це все вище перераховане робить розгортання додатків надзвичайно швидким, рис 3.3. Продуктивність створеної програми залежить від характеристик продукту. Звичайно, для числових алгоритму, який виконує звичайну арифметику всього числа в циклі, незалежно від мови, в якій вона написана. Але якщо взяти більш

складний додаток, то приріст продуктивності може бути неймовірним . Одним з недоліків Python, якщо порівнювати з іншими традиційними мовами програмування, є те, що це не зовсім компільована мова; натомість він частково передає програму у внутрішній формат байтового коду, і цей байтовий код запускається інтерпретатором Python. Однак у довгостроковій перспективі комп'ютери які ми маємо на даний момент мають стільки невикористаного комп'ютерного потенціалу, що для майже 9 з 10 додатків швидкодія зв’язана з вибором мови. Java також може компілюватися в байт-код, але в більшості випадків вона працює повільніше, ніж Python. Крім того, дуже легко поєднувати Python з модулями, написані в C або C++, які можуть бути використані для збільшення швидкості застосування в критичних доменах.

## 3.2 Середа розробки Visual Studio Code

В основі роботи Visual Studio Code є блискавичний редактор вихідного коду, ідеально підходить для щоденного використання. Завдяки підтримці сотень мов, код VS допомагає бути незамінно продуктивним із виділенням синтаксису, відповідності дужок, автоматичним відступом, вибором коробки, фрагментами тощо. Інтуїтивно зрозумілі комбінації клавіш, просте налаштування та відображення комбінацій клавіш для клавіатури, що надаються спільнотою, дозволяють легко переміщувати код рис 3.5.

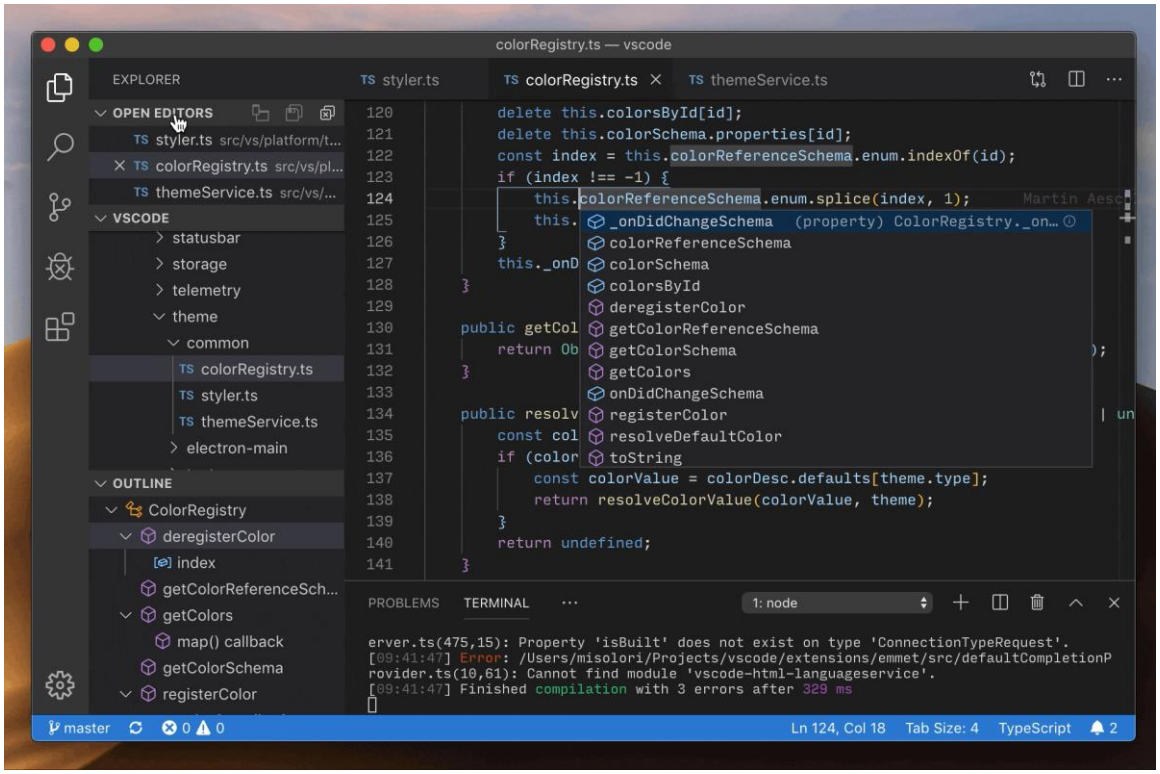



Рисунок 3.5 — Середовище написання

Для серйозного кодування можна користуються інструменти з більш зрозумілим кодом, ніж просто блоки тексту. Код Visual Studio включає вбудовану підтримку для завершення коду IntelliSense, багате розуміння семантичного коду та навігації та рефакторинг коду.

І коли кодування стає жорстким, жорстке стає налагодженням. Налагодження часто є однією з особливостей, яку розробники найбільше пропускають у досвіді коротшого кодування. Visual Studio Code включає інтерактивний відладчик, тому можливо переходити через вихідний код, перевіряти змінні, переглядати стеки викликів та виконувати команди в консолі.

Код VS також інтегрується з інструментами побудови та створення сценаріїв для виконання звичайних завдань, що сприяє більш швидкому щоденному робочому процесу VS Code має підтримку Git, тому можливо працювати з керуванням джерелами, не виходячи з редактора, включаючи перегляд очікуючих змін.

## 3.3 Фреймвок Flask

Flask є веб-фреймворком. Це означає, що Flask надає вам інструменти, бібліотеки та технології, що дозволяють створювати веб-додаток. Цей веб-додаток може бути деякими веб-сторінками, блогом, вікі або настільки ж великим, як веб-додаток для календаря або комерційний веб-сайт.

Flask є частиною категорій мікро-фреймворк. Мікро-фреймворк, як правило, є фреймворками, мало залежними від зовнішніх бібліотек. Це має плюси і мінуси. Плюси полягають у тому, що фреймворки легкі, є невелика залежність для оновлення та спостереження за помилками безпеки. Мінуси полягають у тому, що деякий час вам доведеться робити більше роботи самостійно або збільшувати список залежностей, додаючи плагіни. У випадку з Flask її залежність:

- Werkzeug а WSGI бібліотека утиліт
- jinja2 що являється шаблонізатором

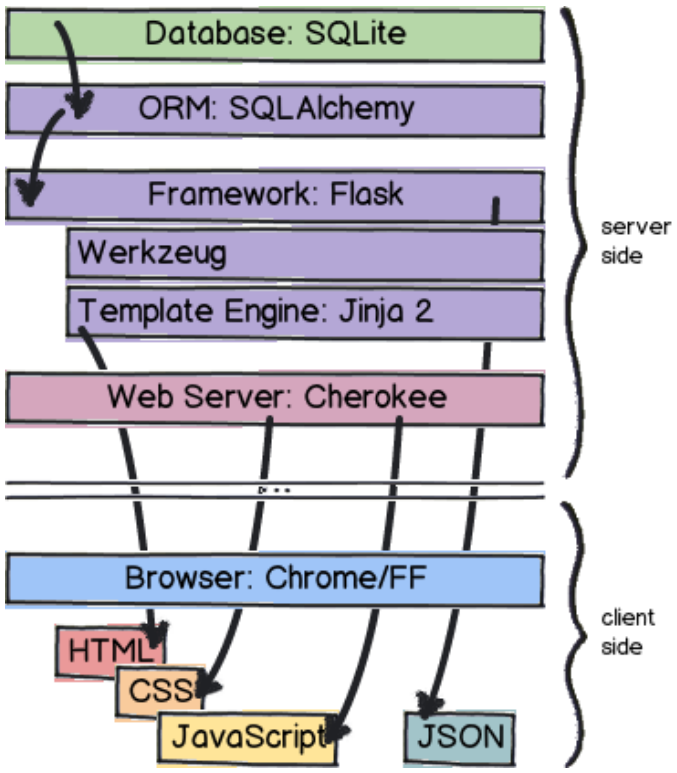



Рисунок 3.6 — Принцип роботи Flask

Тобто Flask - це легка структура веб-додатків WSGI. Він призначений для швидкого та легкого початку роботи з можливістю масштабування до складних програм. Він розпочався як проста обгортка навколо Werkzeug та Jinjai , а потім став однією з найпопулярніших фреймворків для веб-додатків Python. Принцип роботи зображено на рис. 3.6.

Flask дає вибір, але не використовує будь-яких залежностей та макетів проекту. Розробник повинен вибрати інструменти та бібліотеки, які вони хочуть використовувати. Спільнота пропонує багато розширень, які полегшують додавання нових функціональних можливостей.

## 3.4 Flask-SQLAlchemy

Flask-SQLAlchemy - це розширення для Flask, яке додає підтримку SQLAlchemy до вашої програми рис 3.7. Він спрямований на спрощення використання SQLAlchemy з Flask, надаючи корисні параметри за замовчуванням та додаткові помічники, що полегшують виконання спільних завдань.

Рисунок 3.7 — Поєднання Flask та Flask-SQLAlchemy

Зауважимо, що у нас є Flask з розширенням Flask-sqlalchemy, яка спрощує використання SQLAlchemy, надаючи корисні параметри за замовчуванням та додаткових помічників, полегшуючи виконання загальних завдань.

Згідно з веб-сайтом, “SQLAlchemy - це інструментарій Python SQL та Object Relational Mapper, який надає розробникам додатків повну потужність та гнучкість SQL”.

Після прочитання визначення вище, перше питання, що виникає, - це що таке об'єктивний реляційний Mapper. Об'єктний реляційний Mapper, також відомий як ORM, - це техніка, що використовується для запиту запитів до бази даних, використовуючи об'єктно-орієнтовану парадигму бажаної мови (в даному випадку Python).

Ще простіше кажучи, ORM можна розглядати як перекладача, який переводить код з одного набору абстракцій в інший рис 3.8. У нашому випадку - від Python до SQL. Існує дуже багато різних причин використовувати ORM, крім того, що не потрібно створювати рядки SQL. Деякі з яких:

— Прискорюючи розробку веб-сторінок, оскільки нам не потрібно перемикатися між написанням Python та SQL
— Усунення повторюваного коду
— Упорядкування робочого процесу та запит даних більш ефективно
— Відняття системи баз даних таким чином перехід між різними базами даних стає плавним
— Створення кодового коду для основних операцій CRUD

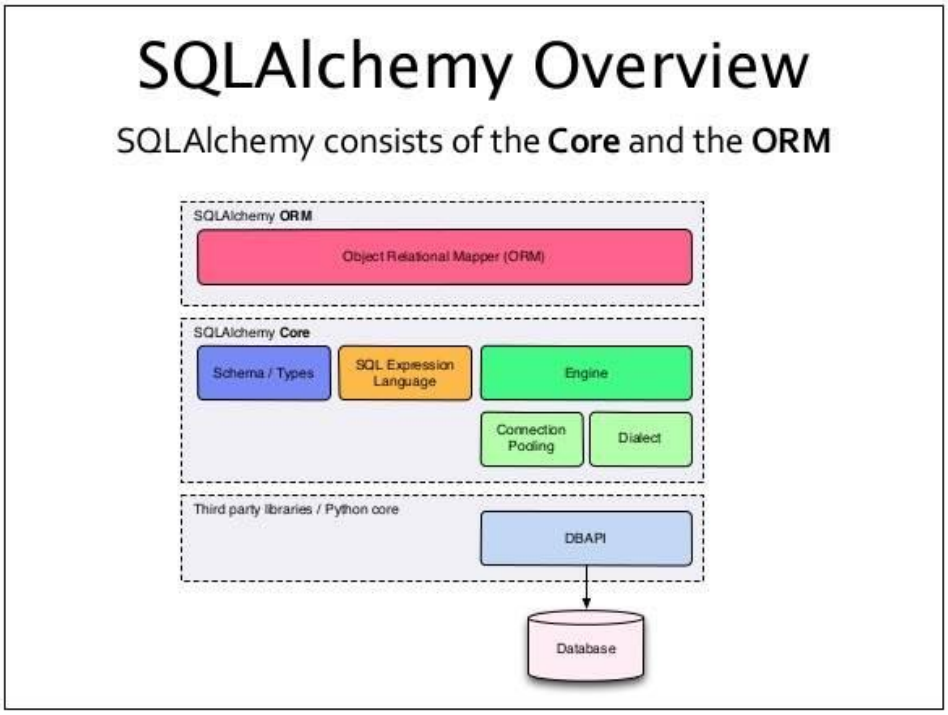


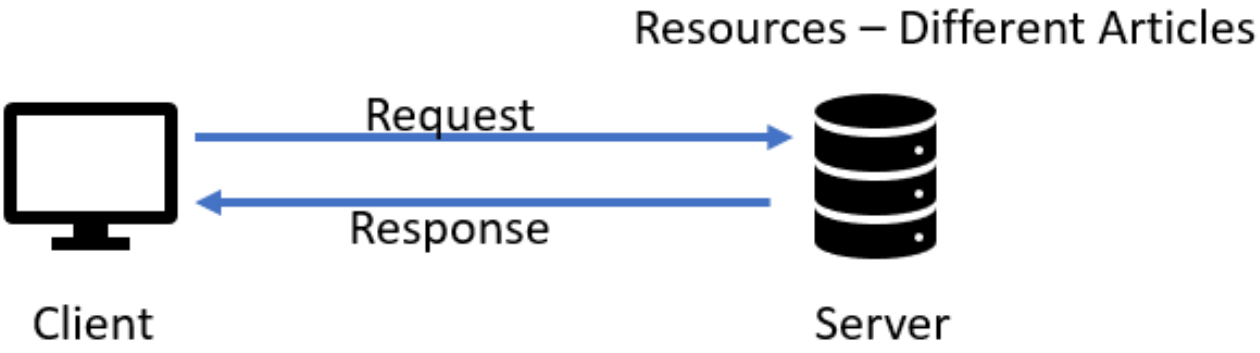

Рисунок 3.8 — ORM Flask-SQLAlchemy

## 3.5 СУБД SQLite

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер рис 3.9, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, виклики функцій використовуються як протокол (API) бібліотеки SQLite. Таким чином зменшуються накладні витрати також спрощується програма і час відгуку.

SQLite працює досить просто зберігаючи в єдиному файлі всю базу даних на комп'ютері на якому працює сама програма. Така простота робиться за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних,

блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.




| | |
|---|---|
| Endpoint | www.medium.com |
| Method | GET, PUT, POST, DELETE, PATCH |
| Header | Authentication information for both client and server |
| Body | Any information to be sent to server |

Рисунок 3.9 — База даних

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

Разом із комплектом постачання працює також частина функціоналу яка відповідає за клієнтів у вигляді виконуваного файлу sqlite3, за його допомогою можна

побачити основні функції бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі рушія можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

В даному випадку база даних відіграє важливу роль так як потрібно десь зберігати інформацію про користувачів та замовлення стислий опис роботи додатку зображений на рис. 3.10.

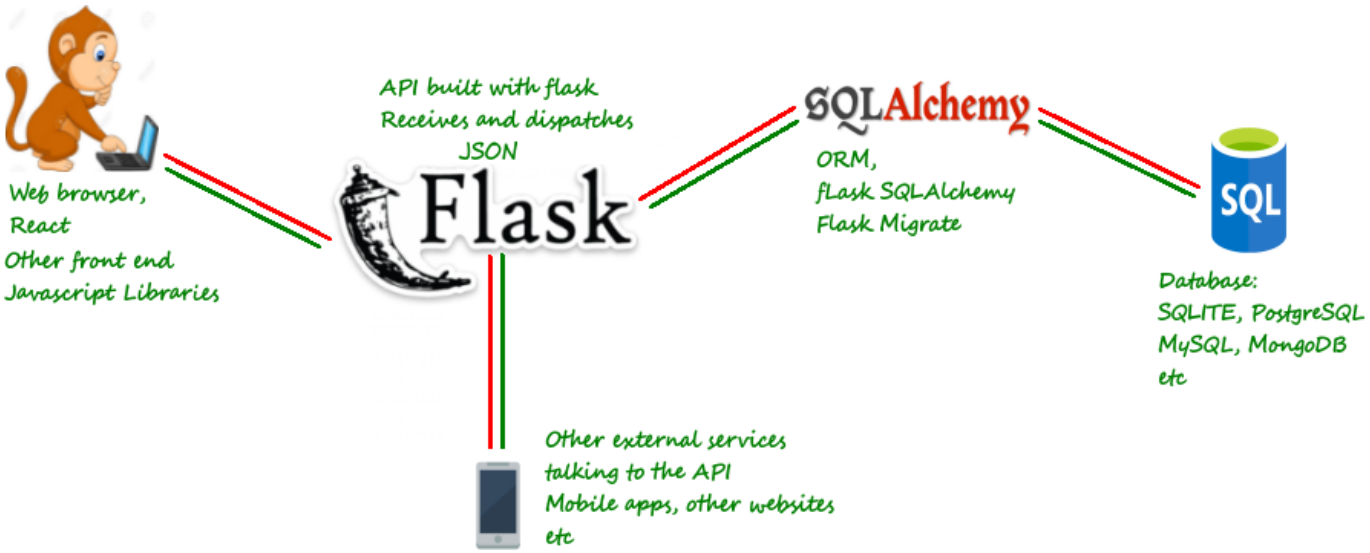



Рисунок 3.10 — Принцип роботи додатку

## 3.6 Хмарна платформа Heroku

Heroku - хмарна платформа на основі контейнерів як послуга (PaaS). Розробники використовують Heroku для розгортання, управління та масштабування сучасних додатків. Ця платформа елегантна, гнучка і проста у використанні, пропонуючи розробникам найпростіший шлях до виходу їхніх програм на ринок.

Heroku повністю керується, даючи розробникам свободу зосереджуватися на своєму основному продукті, не відволікаючись на обслуговування серверів,

апаратних засобів чи інфраструктури. Досвід Heroku надає послуги, інструменти, робочі процеси та підтримку поліглотів - усі вони розроблені для підвищення продуктивності розробника рис 3.11.


Рисунок 3.11 — Логотип Heroku

Heroku відомий тим, що запускає програми в dynos - це справді лише віртуальні комп’ютери, які можуть отримувати більше або менше ресурсів, виходячи з того, наскільки великий ваш додаток. Подумайте про dynos як проі будівельні блоки для запуску вашої програми.

Якщо ви хочете обробити більше даних або виконати більш складні завдання, вам потрібно буде додати більше блоків (що називається масштабування по горизонталі) або збільшити розмір блоків (те, що називається масштабуванням по вертикалі). Потім Heroku стягує з вас щомісячну плату виходячи з кількості наявних вами “диносів” та розміру кожного “дино”.

Хоча Heroku стягує з вас платню, вони насправді не приймають ваш додаток. Насправді вся платформа Heroku, а також кожен додаток, побудований на Heroku, розміщені на веб-службах Amazon (AWS).

Це ставить питання –

Навіщо використовувати Heroku, коли AWS присутній?

Щоб зрозуміти, що таке Heroku, ми почнемо з того, що що не являється Heroku. Можливо, ви зрозуміли, що Heroku працює на веб-службах Amazon (AWS), і тепер ви запитуєте себе, чому ви не розгортаєтесь до AWS і повністю обходите Heroku. Перш за все, Heroku та AWS - це не одне і те ж.

AWS - інфраструктура як постачальник послуг (IaaS), тобто вони відповідають за управління великими центрами обробки даних. Ці центри обробки даних називаємо "хмарою". Такі компанії, як AWS, Azure та Google, створили IaaS, щоб розробники могли платити за розміщення своїх додатків у цих центрах обробки даних, а не для створення самих серверів.

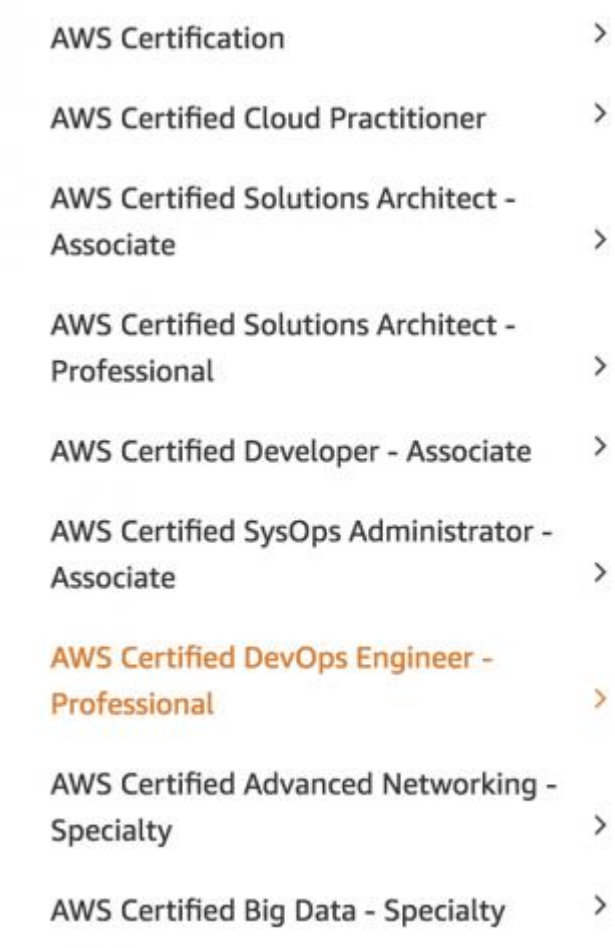



Рисунок 3.12 — Принцип роботи

Це великий компроміс, але через характер їхнього бізнесу постачальники IaaS більше переймаються роботою центрів обробки даних, ніж досвід розробників у роботі з ними. Це означає, що для забезпечення роботи програми, особливо в масштабах, потрібен високий рівень знань про AWS рис 3.12.

Якщо будь-яка міра того, скільки потрібно знати, AWS пропонує 8 різних сертифікацій для того, щоб люди підтвердили свої знання.

Heroku, з іншого боку, - це Платформа як Служба, яка базується на AWS, щоб забезпечити досвід, розроблений спеціально для полегшення життя розробників. Наприклад, для того, щоб програма не працювала в масштабі на Heroku, потрібно лише знання кількох команд на Heroku CLI та панелі керування. Ці команди легко знайти в документації Heroku.

Чому ще люди обирають Heroku?

Це найкращий досвід для розробників класів.

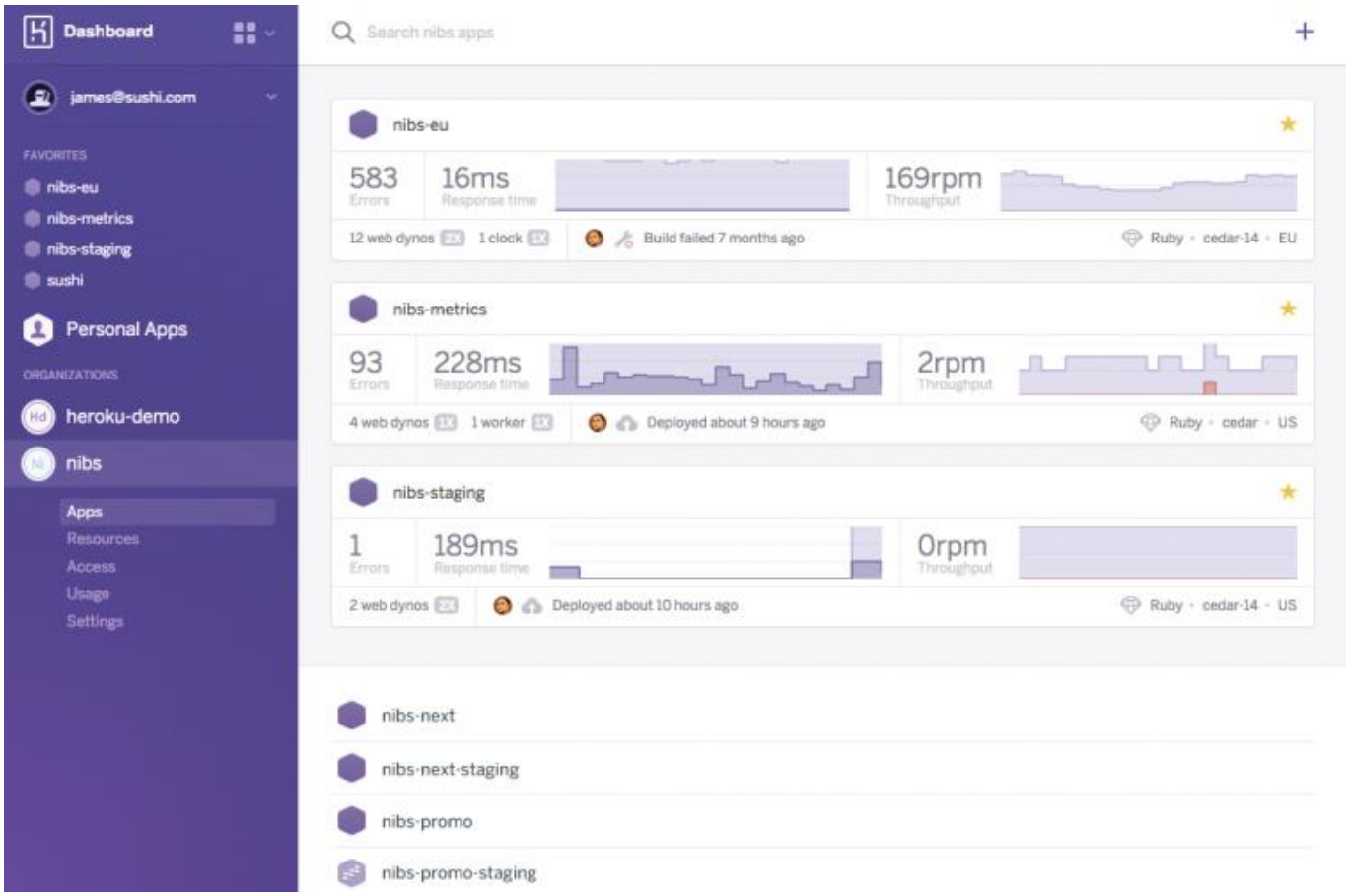



Рисунок 3.13 — Принцип роботи

І знову Heroku створили розробники для розробників. За досвідом легко орієнтуватися, розробники точно знають, що їм потрібно робити під час входу, і вони точно знають, як їх додаток працює щосекунди на платформі рис 3.13.

Це відкрито і розширюється

Heroku є відкритим і розширюваним, тому розробники можуть створювати будь-яку мову, яку вони оберуть. Будь це Nodejs, Ruby, PHP, Python, Java, це не має значення.

Не тільки розробники можуть створювати будь-яку мову, яку ви вибрали, але існує величезна мережа додатків Heroku. Додатки Heroku - це потужні функції та функціональні можливості, які можна розгортати у вашому додатку одним натисканням кнопки. Деякі з цих заявок платні, а деякі безкоштовні, але вони можуть поголити сотні годин від вашого наступного проекту.

Розробники додатків покладаються на програмні абстракції для спрощення розробки та підвищення продуктивності. Що стосується запущених додатків, то контейнеризація знімає тягар управління апаратними або віртуальними машинами. Замість управління апаратом ви розгортаєте додаток у Heroku, який пакує код і залежність програми в контейнери - легкі, ізольовані середовища, які забезпечують обчислення, пам'ять, ОС та ефемерну файлову систему. Контейнери, як правило, працюють на спільному хості, але вони повністю ізольовані один від одного.

Платформа Heroku використовує модель контейнера для запуску та масштабування всіх додатків Heroku. Контейнери, які використовуються в Heroku, називають "диносом". Dynos є ізольованими, віртуалізованими контейнерами Linux, призначеними для виконання коду на основі визначеної користувачем команди. Ваш додаток може масштабувати до будь-якої заданої кількості диносів залежно від потреб у ресурсі. Можливості управління контейнерами Heroku надають вам простий спосіб масштабувати та керувати кількістю, розміром та типом “диносів”, які може знадобитися вашому додатку в будь-який момент.

## 3.7 Висновки

Для створення продукту було використано бібліотеку TeleBotAPI яка значно полегшила роботу та надала змогу з’єднання бота/підсистеми до платіжної системи оплати послуг LiqPay.

За допомогою Flask-SQLAlchemy було зроблено реляційну базу даних для товарів. Завдяки SQLite адміністратор чи група адміністраторів може легко добавляти і видаляти будь-який товар, оперативно працювати з замовленнями та базою даних в цілому, не покидаючи підсистему. Також, адміністратори можуть повністю видалити та створити нову базу даних.

Сам проект було розміщено на сервісі Heroku, хмарна платформа яка забезпечуэ бездоганну та безперервну роботу підсистеми в онлайн режимі.

# 4 ОПИС РЕАЛІЗАЦІЇ

Для автоматизації продажу товарів, потрібно створити підсистему, яка буде надавати інформацію про кожен товар та в якій можна буде замовити і оплатити онлайн.

Вибір веб-архітектури обумовлений тим, що дана архітектура дозволяє організовувати систему, яка матиме централізоване сховище інформації, що необхідно для збереження та подальшої обробки отриманих даних. Клієнт має можливість вільно пересуватися по пунктам меню структура якого є: Головне меню => Вибір товару => Вибір асортименту => Деталі замовлення (Не обов’язково) => Вага/Кількість => Оплата.

Вибір розподіленої бази даних обумовлений тим, що такий архітектурний підхід забезпечує гнучкість налаштувань кожного окремого вузла (локальної бази даних), збільшує швидкість пошуку даних, адже, в кожній локальній базі зберігаються властиві їй дані та збільшується надійність зберігання даних, адже зменшується вірогідність несанкціонованої зміни даних і також  не малу частину відіграє простота.

## 4.1 Структура програмного забезпечення

Для реалізації задачі був реалізований телеграм бот, сервер якого знаходиться в мережі. На рисунку 4.1 представлений інтерфейс  головного вікна при замовленні товару.


Рисунок 4.1 — Головне вікно додатку

Структура проекту складається з наступних файлів: config, markups, model, payment, main і також папок static де знаходяться картинки товарів та utils де знаходяться всі допоміжні функції як показано на рисунку 4.2

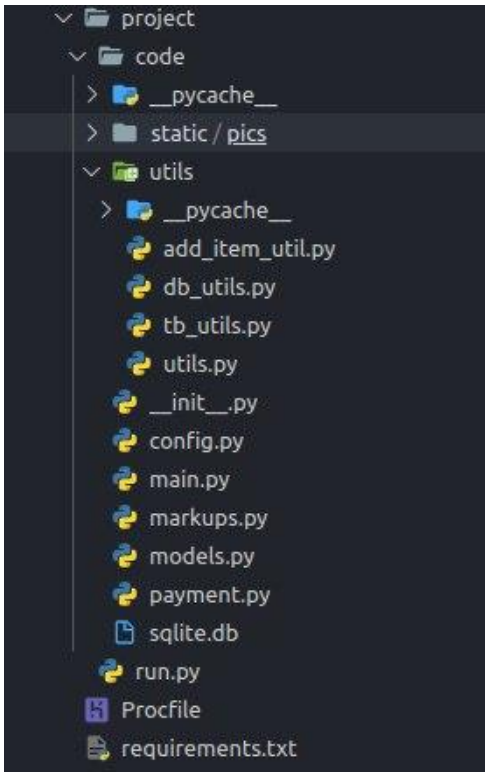



Рисунок 4.2 — Структура файлів проекту

- Main.py — основа бота
- Config.py — всі необхідні конфігурації
- Models.py — створення класів бази даних
- Payment.py — обробка оплати
- Sqlite.db — саме тут зберігається вся інформація про товари це і є база даних даного телеграм бота.

Файл run.py запускає підсистему

## 4.2 Адмінська частина

Також, є можливість безпосередньо через сам телеграм бот добавляти і видаляти товари, категорії та працювати з БД. Це все можна зробити перейшовши в меню адміністратора. Функція доступна тільки якщо користувач є в базі даних адміністраторів. (Рисунок 4.3.)

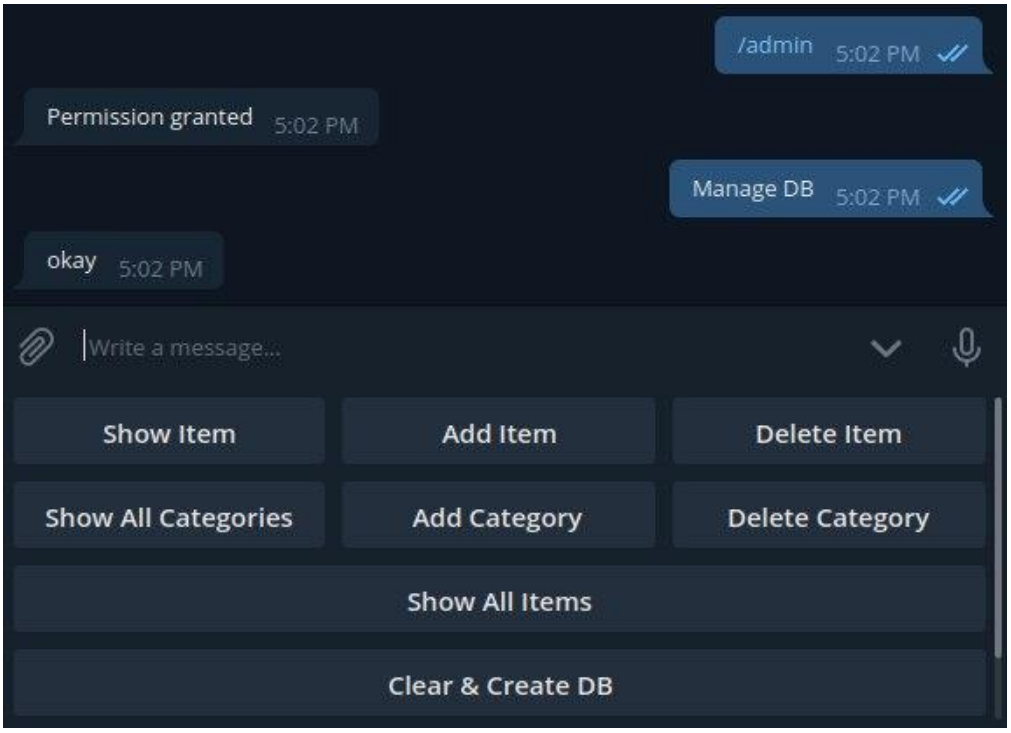



Рисунок 4.3 – Адмін Меню

## 4.3 Висновки

Завдяки великому обсягу документації від Telegram API та TelebotAPI робота над проектом значно полегшилася

Даний телеграм бот працює швидко на віддаленому сервері.

Створення бази даних дало мені чіткі уявлення про збереження інформації про просторові об’єкти у базах даних, їх відношення.

# 5 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Підсистема оплати послуг чи бот в Телеграмі, була створена в додатку Telegram. Тому, користувач може здійснити вхід за допомогою мобільного додатку чи додатку на ПК.

Для відкриття даної підсистеми користувачеві потрібно ввести «@» і назву бота в пошуковому полі чи перейти за посиланням.

## 5.1 Робота з Ботом

Для запуску бота необхідно здійснити клік по функції Старт чи ввести в чат «/start». Після чого користувач зможе побачити асортимент товарі та вибрати потрібну чи цікаву йому категорію, які існують в БД (Рисунок 5.1).

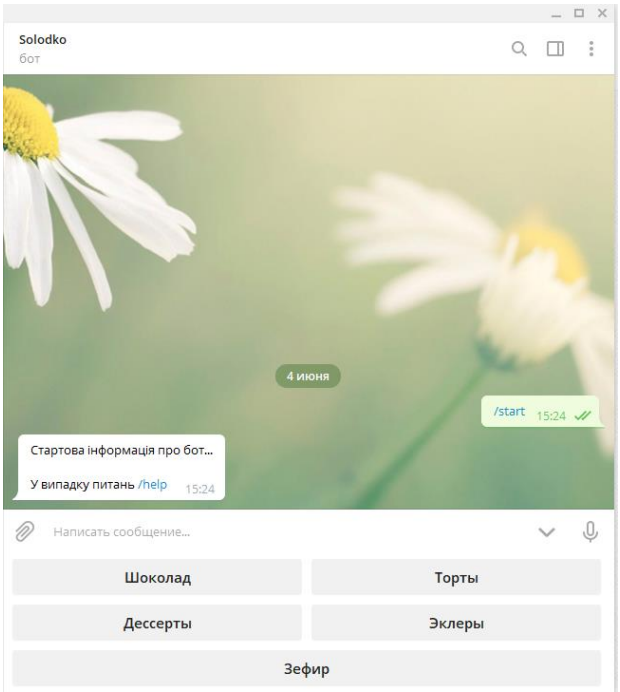



Рисунок 5.1 — Початок

Обравши категорію користувач може використати кнопку для переходу в неї та подальшого вибору товару. (Рисунок 5.2)

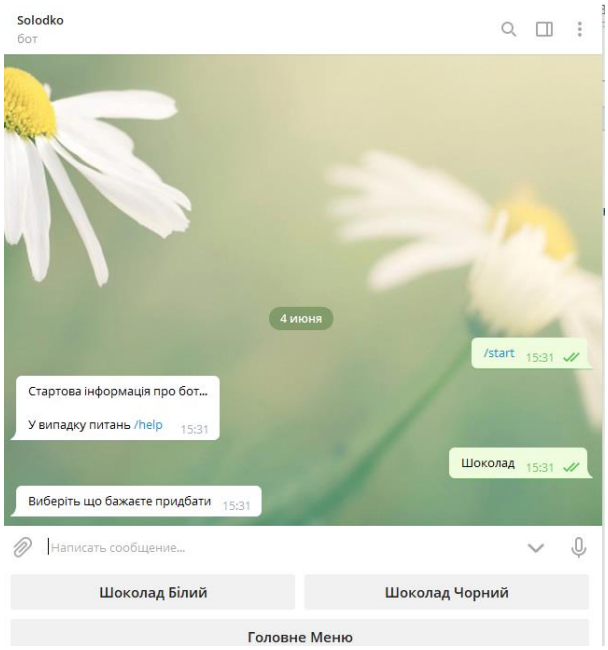



Рисунок 5.2 — Меню в категоріях

Після вибору товару, здійснюється перехід до можливої варіації/типів товару. (Рисунок 5.3.)

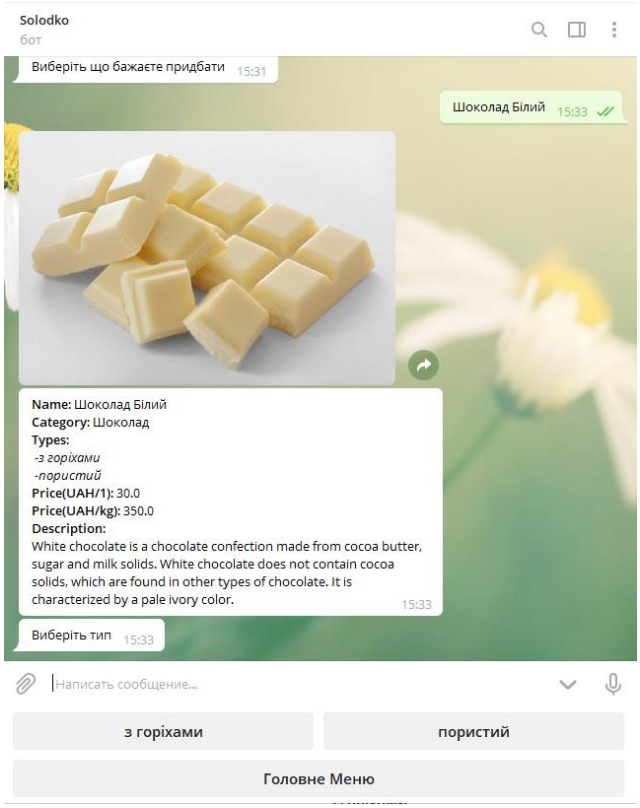



Рисунок 5.3 — Вибір конкретного типу товару

Після вибору типу, користувач може звязатись з компанією чи замовити вказавши кількість чи вагу. Після чого підсистема виводить повністю всі деталі замовлення, його вартість та пропонує здійснити оплату. (Рисунок 5.4.)

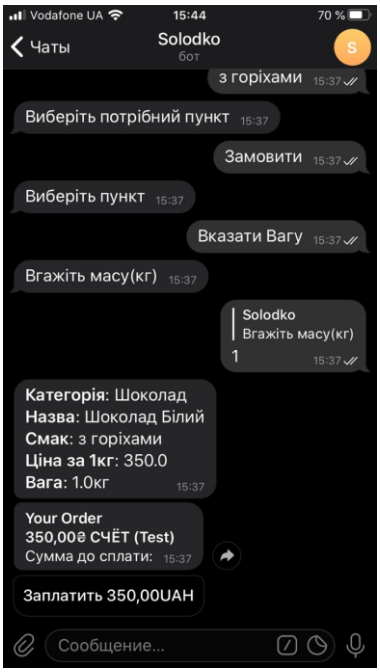



Рисунок 5.4 — Уточнення замовлення та його вивід користувачу

Після цього, користувачу потрібно перейти до сплати, а саме вказати своє ім’я, контактний номер телефону та перейти до способу сплати(Рисунок 5.5.)

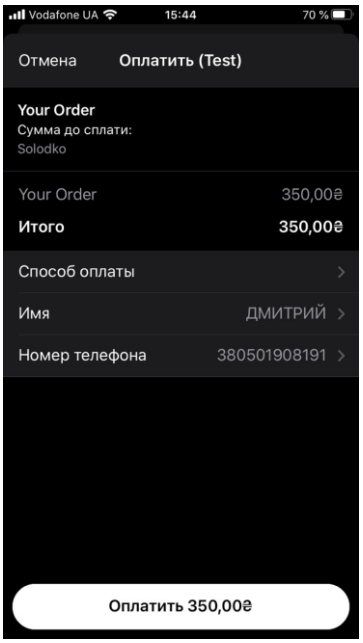



Рисунок 5.5 — Заповнення замовлення

Після кліку на спосіб оплати, користувач може додати нову карту для сплати ввівши її дані чи використати вже збережену. (Рисунок 5.6.)

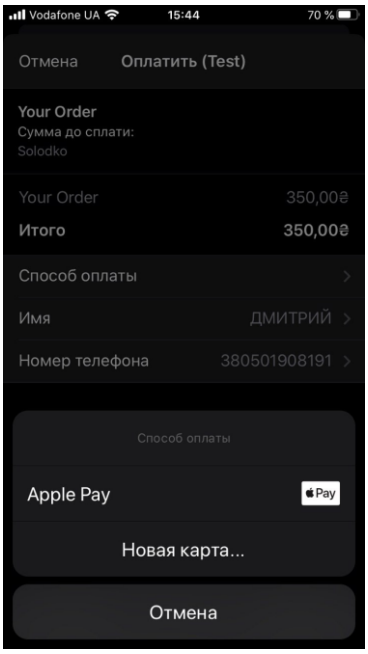



Рисунок 5.6 — Вибір картки

Додати нову карту можливо через введення її даних чи через Приват24. (Рисунок 5.7.)

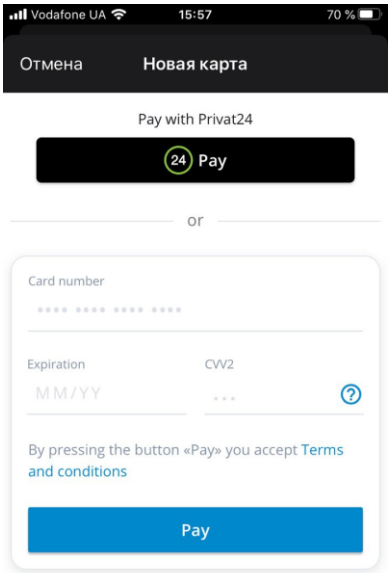



Рисунок 5.7 — Додавання нової карти

Після оплати, користувачу надходить повідомлення про успішну сплату та у нього є можливість переглянути квитанцію. (Рисунок 5.8.)

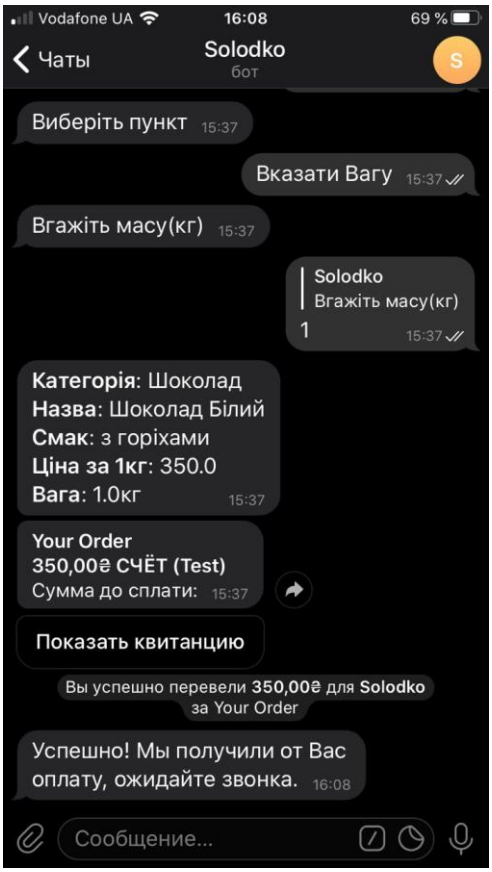



Рисунок 5.8 — Успішна сплата

## 5.2 Робота адміністратора/власника з підсистемою

В підсистемі розроблене окреме меню для власників бота за допомогою якого, зручно користуватися ботом, БД та переглядати данні по замовленням та наявності товару. Доступ надається тільки розробником бота. Для того щоб перейти в режим керування підсистемою, необхідно ввести команду «/admin» - система перевіряє користувача що вводить запит, а саме його ім’я та номер телефону, при успішному підтвердженні бот виводить адмін-меню. (Рисунок 5.9.)

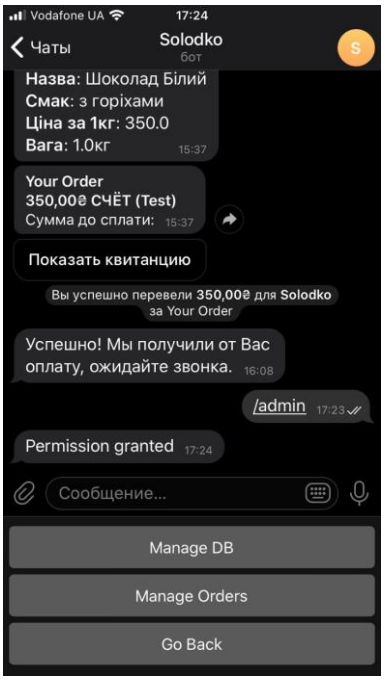



Рисунок 5.9 — Підтвердження доступу до Адмін-меню

Після цього потрібно вибрати наступну дію, перейти до асортименту для редагування, додавання чи видалення товарів/послуг чи перейти до замовлень. При переході до замовлень можна переглянути замовлення чи видалити вже опрацьовані (Рисунок 5.10. - 5.11.)

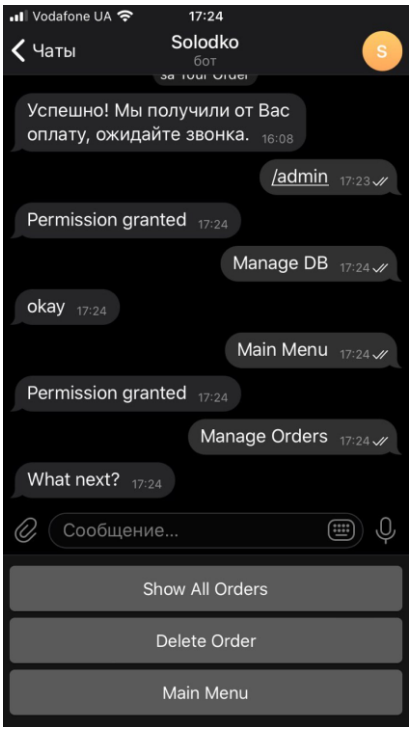



Рисунок 5.10 — Меню Замовлень

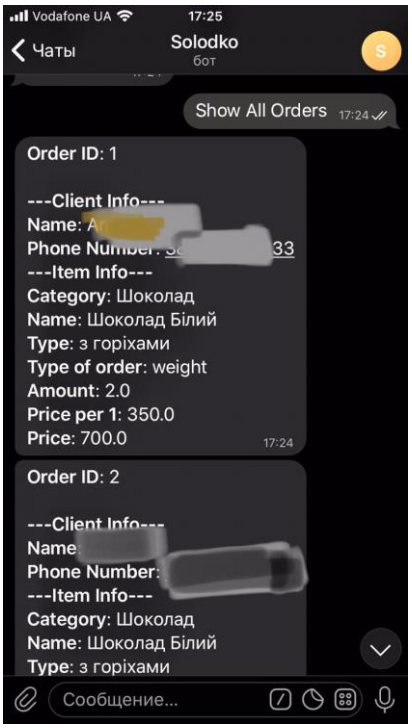



Рисунок 5.11 — Неопрацьовані замовлення

Перейшовши до управління БД, адміністратор має змогу переглянути всі товари, переглянути певну категорії чи товар, додати чи видалити їх. Або повністю видалити і створити нову БД (Рисунок 5.12.)

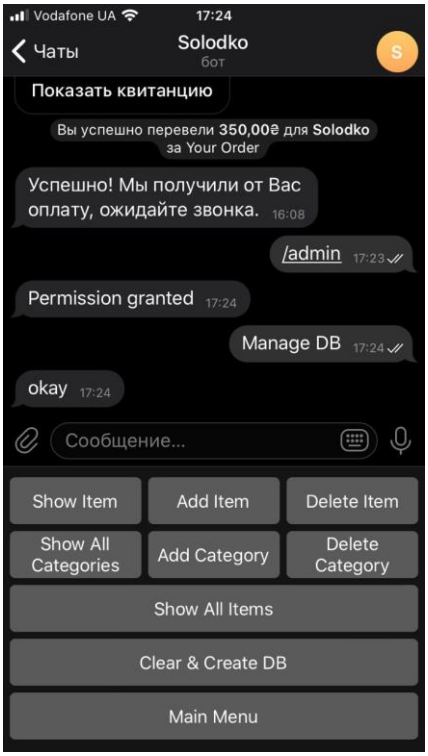



Рисунок 5.12 — Управління БД

## 5.3 Висновки

Розроблена підсистема дуже легка в використанні, не потребує додаткових завантажень, навичок чи знань, як від користувачів, так і для адміна. Підсистема чудово виконує функції роботи з БД, завдяки СКБД та простому функціоналу та функції транзакції завдяки підключенню до платіжної системи.

Попри свою легкість та простоту, система ідеально вирішує проблему СМБ, щодо просування та підключення продаж в месенджері Телеграм.

# 6 ВИСНОВОК

У ході виконаної роботи був отриманий неосяжний досвід роботи з платіжними системами, підсистемами, БД та СКБД. Відповідно до завдання було розроблено платіжну підсистему для оплати товару чи послуг за допомогою ПК чи мобільного пристрою і додатку Telegram.

Невід’ємною частиною роботи було вибрати платіжну систему, через яку працює підсистема, через правила та обмежений функціонал де-яких з них, та підключення до платіжної системи ПриватБанку LiqPay.

Особлива увага приділялась створенню зручного середовища користування для клієнтів та адміністратора. Можливості керування БД та сплати не покидаючи бота.

Розроблена підсистема, може бути використана власниками СМБ, індивідуальними підприємцями і навіть великими компаніями. Актуальність неосяжна в період карантину.

# СПИСОК ДЖЕРЕЛ

1. База Даних / Герасимик Іван кафедра АПЕПС // ТЕФ [Електронний ресурс]. – Режим доступу: http://apeps.kpi.ua/shco-take-basa-danykh
2. Вікіпедія // Платіжна Система [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%96%D0%B6%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0
3. Що таке Python [Електронний ресурс]. – Режим доступу: http://www.plug.org.ua/documentation/about-python
4. Порівняння мов програмування з Python [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/pajtons123/session-1

# ДОДАТОК 1

Підсистема оплати послуг

Текст програми

Аркушів 6

Київ – 2020

```
from code import bot
from code.utils.add_item_util import add_item_util
# markups
from code.markups import (
    home_markup, items_markup, before_order_markup,
    admin_menu_markup, manage_db_markup, delete_DB_procced,
    types_markup, piece_or_weight_markup, manage_orders_markup)
# config
from code.config import (
    commands, messages, ORDER_BUFFER)
# tb_utils
from code.utils.tb_utils import (
    question, check_admin, get_order_weight, show_by_id_item,
    delete_by_id_item, get_cat_name_tg, delete_by_id_cat,
    get_order_piece, get_new_admin, delete_order_tb, send_photo)
# db_utils
from code.utils.db_utils import (
    get_all_items, get_all_categories, is_in_cat,
    get_cat_by_name, is_in_item, get_item_by_name, is_in_type,
    clear_create, get_all_orders)
# utils
from code.utils.utils import (
    print_item_eng, print_category_eng, save_order, print_order_eng)


# Callbacks
@bot.callback_query_handler(func=lambda call:
call.data.startswith('procced'))
def test_callback(call):
    chat_id = call.message.chat.id
    if call.data == "procced_delete":
        clear_create()
        bot.edit_message_text('Done!', chat_id, call.message.message_id)
        bot.send_message(chat_id, 'What next?',
                         reply_markup=manage_db_markup())
    if call.data == "procced_no_detele":
        bot.edit_message_text('OK', chat_id, call.message.message_id)
        bot.send_message(chat_id, 'Then choose something else',
                         reply_markup=manage_db_markup())


# Commands handlers
@bot.message_handler(commands=['start'])
def send_welcome(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, messages['start_mess'],
                     reply_markup=home_markup())


@bot.message_handler(commands=['help'])
```

```
def help_message(message):
    chat_id = message.chat.id
    text = messages['help_mess'] + '\n'
    for comm, info in commands.items():
        text += '/' + comm + '  -    ' + info + '\n'
    bot.send_message(chat_id, text)
    go_home(message)


@bot.message_handler(func=lambda message: message.text == 'Головне Меню')
@bot.message_handler(commands=['home'])
def go_home(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, messages['home'],
                     reply_markup=home_markup())


# ReplyKeyboardMarkup handlers DB
@bot.message_handler(func=lambda message: is_in_cat(message.text))
def items_menu(message):
    text = message.text
    ORDER_BUFFER['category'] = text
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Виберіть що бажаєте придбати',
                     reply_markup=items_markup(get_cat_by_name(text).items))


@bot.message_handler(func=lambda message: is_in_item(message.text))
def types_menu(message):
    text = message.text
    ORDER_BUFFER['type'] = text
    chat_id = message.chat.id
    send_photo(chat_id, text)
    bot.send_message(chat_id, print_item_eng(get_item_by_name(text),
tp='user'),
                     parse_mode='markdown')
    bot.send_message(chat_id, 'Виберіть тип',
                     reply_markup=types_markup(get_item_by_name(text).types))


@bot.message_handler(func=lambda message: is_in_type(message.text))
def before_order_menu(message):
    chat_id = message.chat.id
    ORDER_BUFFER['flavor'] = message.text
    bot.send_message(chat_id, 'Виберіть потрібний пункт',
                     reply_markup=before_order_markup())


# ReplyKeyboardMarkup handlers written
@bot.message_handler(func=lambda message:
```

```
                    message.text == 'Уточнити деталі замовлення')
def order_details_menu(message):
    chat_id = message.chat.id
    bot.send_message(
        chat_id, 'Деталі замовлення можна уточнити в @ReaLKLiM')


@bot.message_handler(func=lambda message:
                    message.text == 'Замовити')
def piece_weight_menu(message):
    chat_id = message.chat.id
    item = get_item_by_name(ORDER_BUFFER['type'])
    if item.price_piece is not None and item.price_weight is not None:
        bot.send_message(chat_id, 'Виберіть пункт',
                         reply_markup=piece_or_weight_markup())
    elif item.price_piece is not None:
        order_piece_menu(message)
    else:
        order_weight_menu(message)


@bot.message_handler(func=lambda message:
                    message.text == 'Вказати Вагу')
def order_weight_menu(message):
    chat_id = message.chat.id
    question(chat_id, 'Вгажіть масу(кг)', get_order_weight)


@bot.message_handler(func=lambda message:
                    message.text == 'Вказати Кількість')
def order_piece_menu(message):
    chat_id = message.chat.id
    question(chat_id, 'Вгажіть кількість', get_order_piece)


# Admin Menu
@bot.message_handler(commands=['admin'])
@check_admin
def admin_menu(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Permission granted',
                     reply_markup=admin_menu_markup())


@bot.message_handler(func=lambda message: message.text == 'Manage DB')
@check_admin
def manage_db(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'okay',
                     reply_markup=manage_db_markup())
```

```
@bot.message_handler(func=lambda message: message.text == 'Add Admin')
@check_admin
def set_new_admin(message):
    chat_id = message.chat.id
    question(chat_id, 'Send me tag', get_new_admin, True)


@bot.message_handler(func=lambda message: message.text == 'Go Back')
@check_admin
def go_back(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'okay',
                     reply_markup=home_markup())


@bot.message_handler(func=lambda message: message.text == 'Main Menu')
@check_admin
def goto_mn(message):
    admin_menu(message)


@bot.message_handler(func=lambda message: message.text == 'Clear & Create
DB')
@check_admin
def clear_delete_bd(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'This cannot be undone.\nAre you sure?',
                     reply_markup=delete_DB_procced())


@bot.message_handler(func=lambda message: message.text == 'Add Item')
@check_admin
def ask_item(message):
    chat_id = message.chat.id
    add_item_util(chat_id)


@bot.message_handler(func=lambda message: message.text == 'Show Item')
@check_admin
def Show_item(message):
    chat_id = message.chat.id
    question(chat_id, 'Enter item ID', show_by_id_item)


@bot.message_handler(func=lambda message: message.text == 'Delete Item')
@check_admin
def Delete_item(message):
    chat_id = message.chat.id
```

```
    question(chat_id, 'Enter item ID', delete_by_id_item)


@bot.message_handler(func=lambda message: message.text == 'Show All Items')
@check_admin
def show_all_items(message):
    chat_id = message.chat.id
    for item in get_all_items():
        bot.send_message(chat_id, print_item_eng(item),
parse_mode='markdown')


@bot.message_handler(func=lambda message: message.text == 'Add Category')
@check_admin
def ask_category(message):
    chat_id = message.chat.id
    question(chat_id, 'Enter Category Name', get_cat_name_tg)


@bot.message_handler(func=lambda message: message.text == 'Show All
Categories')
@check_admin
def ask_category(message):
    chat_id = message.chat.id
    for category in get_all_categories():
        bot.send_message(chat_id, print_category_eng(category),
                         reply_markup=manage_db_markup(),
parse_mode='markdown')


@bot.message_handler(func=lambda message: message.text == 'Delete Category')
@check_admin
def delete_category(message):
    chat_id = message.chat.id
    question(chat_id, 'Enter Category Name', delete_by_id_cat)


@bot.message_handler(func=lambda message: message.text == 'Manage Orders')
@check_admin
def manage_orders(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'What next?',
                     reply_markup=manage_orders_markup())


@bot.message_handler(func=lambda message: message.text == 'Delete Order')
@check_admin
def delete_order(message):
    chat_id = message.chat.id
    question(chat_id, "Type order's ID", delete_order_tb)
```

```
@bot.message_handler(func=lambda message: message.text == 'Show All Orders')
@check_admin
def show_all_orders(message):
    chat_id = message.chat.id
    if len(get_all_orders()) == 0:
        bot.send_message(chat_id, "There are no orders")
    for order in get_all_orders():
        bot.send_message(chat_id, print_order_eng(
            order), parse_mode='markdown')


# Payment
@bot.shipping_query_handler(func=lambda query: True)
def shipping(shipping_query):
    bot.answer_shipping_query(shipping_query.id, ok=True,
shipping_options=shipping_options,
                              error_message='Oh, seems like our Dog couriers
are having a lunch right now. Try again later!')


@bot.pre_checkout_query_handler(func=lambda query: True)
def checkout(pre_checkout_query):
    bot.answer_pre_checkout_query(pre_checkout_query.id, ok=True,
                                  error_message="Aliens tried to steal your
card's CVV, but we successfully protected your credentials,"
                                                " try to pay again in a few
minutes, we need a small rest.")


@bot.message_handler(content_types=['successful_payment'])
def got_payment(message):
    save_order(message.successful_payment.order_info)
    bot.send_message(message.chat.id,
                     'Успешно! Мы получили от Вас оплату, ожидайте звонка.')
    if message.chat.id != 399033754:
             bot.send_message(399033754, 'You have new order')
```